四川師範大學
SICHUAN NORMAL UNIVERSITY

第 4 章　　串

DATA STRUCTURE

计算机科学学院　廖雪花

本章内容简介

串

- 4.1 串类型的定义
- 4.2 串的表示和实现
- 4.3 串的模式匹配算法

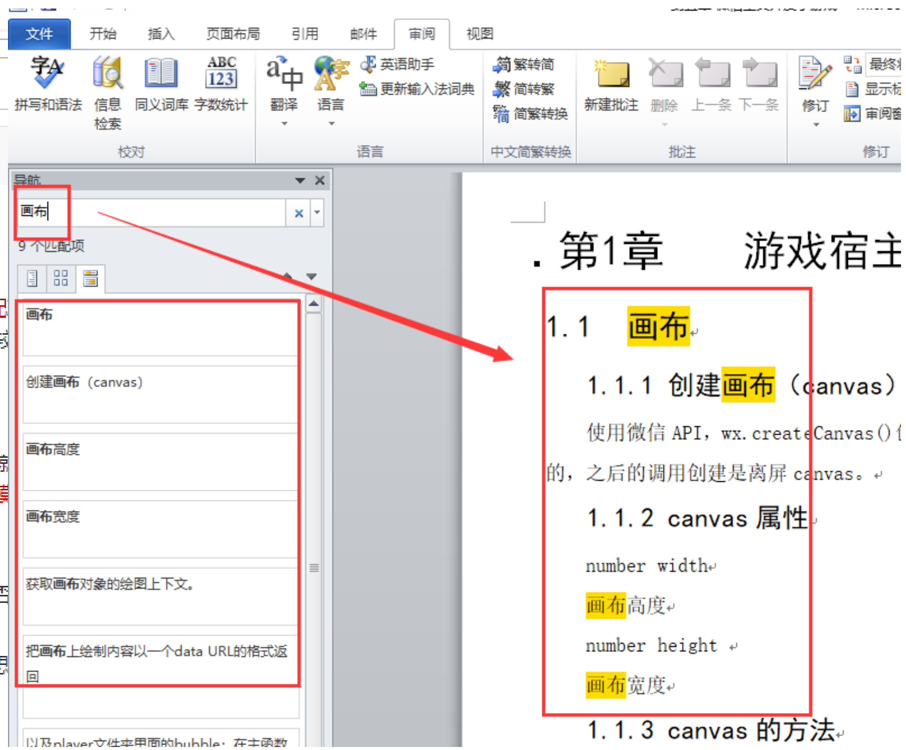
4.1 串的模式匹配算法

廖雪花　LiaoXuehua

SICHUAN NORMAL UNIVERSITY

知识回顾

- 这是串的一种重要操作，很多软件，若有“编辑”菜单项的话，则其中必有“查找”子菜单项。



知识回顾

- 串匹配(查找)的定义:

  INDEX (S, T, pos)

- 初始条件：串S和T存在，T是非空串，1≤pos≤StrLength(S)。
- 操作结果：若主串S中存在和串T值相同的子串返回它在主串S中第pos个字符之后第一次出现的位置；否则函数值为0。

BF（Brute-Force，暴力匹配）算法

- 最基本的字符串匹配算法。
- 将目标串S的第一个字符与模式串T的第一个字符进行匹配，若相等，则继续比较S的第二个字符和T的第二个字符；若不相等，则比较S的第二个字符和T的第一个字符，依次比较，直到得出最后的匹配结果。

BF（Brute-Force，暴力匹配）算法

1趟	a	b	a	b	c	a	b	c	a	c	b	a	b
	a	b	c										

2趟	a	b	a	b	c	a	b	c	a	c	b	a	b
		a											

3趟	a	b	a	b	c	a	b	c	a	c	b	a	b
			a	b	c	a	c						

BF（Brute-Force，暴力匹配）算法

4 趟 a b a b c a b c a c b a b

a

5 趟 a b a b c a b c a c b a b

a

6 趟 a b a b c a b c a c b a b

a b c a c

BF（Brute-Force，暴力匹配）算法

- 下面以定长的顺序串类型作为存储结构，给出具体的串BF匹配算法。

```
int Index(SString S,SString T,int pos){
    i=pos; j=1;
    while (i<=S[0] &&j<=T[0])
      if (s[i]==T[j]) {++i;++j;}
      else {i=i-j+2;j=1;}
    if (j>T[0]) return i-T[0];
    else return 0;
}
```

BF（Brute-Force，暴力匹配）算法

■算法时间复杂度分析

（设主串长度为n，模式串长度为m）

- 最好情况
  - 如：
    - 正文 S="aaaaaabaaaaaaaaaaaaaaa"
    - 模式T="aaaaaab"
  - 时间复杂度为O(m)

BF（Brute-Force，暴力匹配）算法

- 最坏情况:
  - 例如：
    - 正文 S="aaaaaaaaaaaaaaaaaaaaaaaaaaaaab "
    - 模式T="aaaaaab"
  - 比较次数为(n - m + 1)×m。
    - 一般情况下m<<n，所以时间复杂度为O(m*n)。
- 平均情况:
  - 接近O(m+n)，至今仍被使用。

BF（Brute-Force，暴力匹配）算法

BF算法的特点：

- 简单，易于理解，效率较高；
- 算法的时间复杂度O(n*m)。(其中n,m分别为主串和模式串的长度)

有没有改进的算法呢？

模式匹配的改进算法

改进思路

- 当遇到一次$s_i \neq t_j$,主串要回退到i-j+2的位置，而模式串要回到第一个位置（即j=1的位置);
- 但当一次比较出现$s_i \neq t_j$时，则应该有：

$$"s_{i-j+1}s_{i-j+2}\ldots\ldots s_{i-1}"="t_1t_2\ldots\ldots t_{j-2}t_{j-1}"$$

- **改进**：每当一趟匹配过程出现$s_i \neq t_j$时，主串指示器i不用回溯，而是利用已经得到的“部分匹配”结果，将模式串向右“滑动”尽可能远的一段距离后，继续进行比较。

模式匹配的改进算法（KMP算法）

KMP算法

- D.E.Knuth， J.H.Morris， V.R.Pratt
- 仅用O(m+n)时间
  - 发生失配时:
    - 正文串扫描指针i，不回退。
    - 模式串扫描指针j，不一定要回退到头。

模式匹配的改进算法（KMP算法）

■分析与示例：

1趟

a b a b c a b c a c b a b

a b c

2趟

a b a b c a b c a c b a b

a b c a c

3趟

a b a b c a b c a c b a b

a b c a c

模式匹配的改进算法（KMP算法）

■讨论一般情况：

- 设：　主串　　$S=\text{"}s_1s_2\ldots s_i\ldots s_n\text{"}$ ，
  　　模式串　$T=\text{"}p_1p_2\ldots p_j\ldots p_m\text{"}$

- 问：当某趟比较发生“失配”（即$s_i \neq p_j$）时，模式串应该向“右”滑动的可行距离为多长？(不需要回溯i指针）

模式匹配的改进算法（KMP算法）

■讨论一般情况:

- 主　串:　$s_1s_2\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\, s_i\ldots\ldots\ldots\ldots\ldots\ldots\, s_n$
- 模式串:　$t_1\ldots t_{k-1}t_k\ldots t_{j-k+1}\ldots t_{j-1}t_j\ldots\ldots t_m$

$t_1\ \ldots\ t_{k-1}t_k\ldots\ t_j\ldots\ldots t_m$

模式匹配的改进算法（KMP算法）

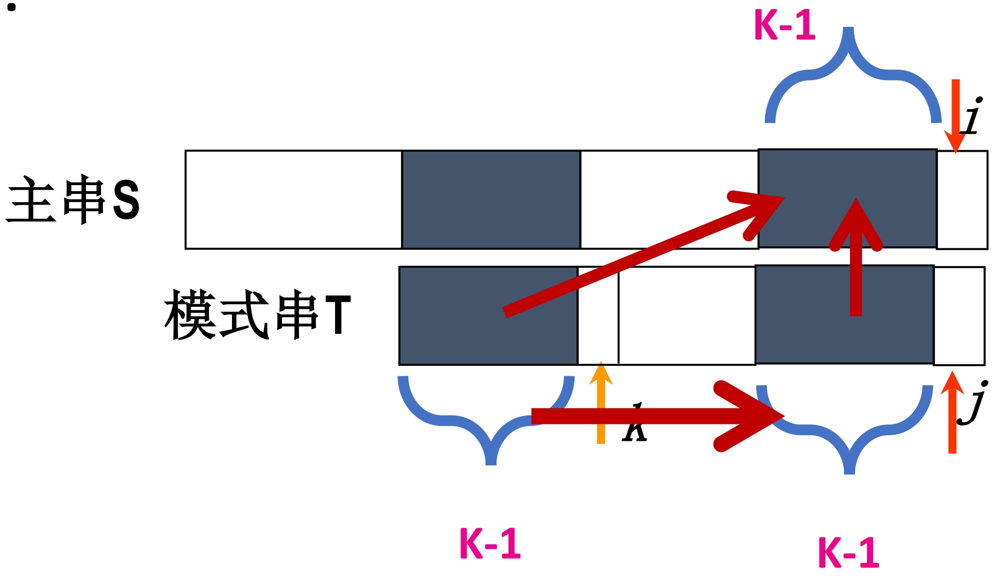
- 解：设某趟匹配发生$s_i \neq p_j$时，s_i应该与p_k（k < j）继续比较。

  根据："$p_1p_2 \ldots p_{k-1}$" = "$s_{i-k+1}s_{i-k+2} \ldots s_{i-1}$"

  "$p_{j-k+1}p_{j-k+2} \ldots p_{j-1}$" = "$s_{i-k+1}s_{i-k+2} \ldots s_{i-1}$"

  可以推出："$p_1p_2 \ldots p_{k-1}$" = "$p_{j-k+1}p_{j-k+2} \ldots p_{j-1}$"

  示意图如下：



模式匹配的改进算法（KMP算法）

- 讨论一般情况：

令next(j) = k

$$
\text{next}(j)=\begin{cases}0 & \text{当}j=1\\ \text{Max}\{k \mid 1<k<j \text{ 且}"p_1p_2\ldots p_{k-1}" = "p_{j-k+1}p_{j-k+2}\ldots p_{j-1}"\} & \text{当此集合非空}\\ 1 & \text{其他情况}\end{cases}
$$

模式匹配的改进算法（KMP算法）

- 例1：计算如下模式串的next函数值。

j	1	2	3	4	5
P_j	a	b	c	a	c
next(j)	0	1	1	1	2

模式匹配的改进算法（KMP算法）

■例2：计算如下模式串的next函数值。

j	1	2	3	4	5	6	7	8
P_j	a	b	a	a	b	c	a	c
next(j)	0	1	1	2	2	3	1	2

KMP算法实现

```
int Index_KMP(SString S,SString T,int pos){
     i=pos;j=1;
     while (i<=S[0]&&j<=T[0]) {
        if (j=0||S[i]==T[j]) {++i,;++j;}
        else  j=next[ j ];
     }
    if (j>T[0]) return i -T[0];
    else return 0;
}// Index_KMP
```

时间复杂度:O(n)

KMP算法实现

■ 模式串的next函数值的求解:

由定义知：next[1]=0,

设next[j]=k,表明"$p_1p_2…p_{k-1}$"= "$p_{j-k+1}p_{j-k+2}…p_{j-1}$ "　　①

（其中1 < k < j, 且不存在k'(k' > k)满足上式）

问：next[j+1]=k'=?

解：　情况1：若$p_K = P_j$,即 "$p_1p_2… p_{k-1}\ p_k$"= "$p_{j-k+1}p_{j-k+2}… p_{j-1}\ p_j$ "　　②

则next[j+1]=next[j]+1=k+1；

情况2：若$p_K \neq P_j$,即"$p_1p_2… p_{k-1}\ p_k$ “≠ "$p_{j-k+1}p_{j-k+2}… p_{j-1}\ p_j$",

但 "$p_1p_2… p_{k-1}$ “= "$p_{j-k+1}p_{j-k+2}… p_{j-1}$ ",

则应将模式向右滑动至模式中的next[k]=k'个字符比较，重复上述过程直至P_j 和模式中的某个字符匹配成功或不存在任何k'(1<k'<j)满足 ②，则令next[j+1]=1。

KMP算法实现

- 模式串的next函数值的求解算法实现:

```
void get_next(SString T,int &next[ ]) {
 i=1;next[1]=0;j=0;
 while (i<T[0]){
  if (j==0 ||T[i]==T[j])
   { ++i;++j;next[i]=j;}
  else j=next[j];
 }
}// get_next
```

KMP算法时间复杂度

- KMP算法的时间复杂度O(m+n)

计算next函数	O(m)
匹配函数Index_KMP	O(n)

KMP算法优化

KMP算法能否进一步优化呢？

- 以字符串”a b a b c”为例，当第二个a失配后，说明被匹配字符串一定不为a，这时候我们可以将当前值与串的sub[next[i]]值进行比较，如果相同，则将nextval[next[i]]值存入nextval[i],不同则将当前值的next的值存入nextval[i];
- 在next数组的基础上实现KMP的优化，请参看教材算法4.8。

串操作应用

- 文本编辑

- 建立词索引表

本节要点

- 串的模式匹配算法：BF算法
  - ✓ 简单、易理解，主串子串都回退
  - ✓ 效率不高，时间复杂度：O(n*m)

- 串的模式匹配算法：KMP算法
  - ✓ 主串不回退，子串回退
  - ✓ 计算next函数值
  - ✓ 效率高，时间复杂度：O(n+m)

SICHUAN NORMAL UNIVERSITY

感谢聆听